第42卷 第5期
2022年10月

桂林电子科技大学学报
Journal of Guilin University of Electronic Technology

Vol. 42,No. 5
Oct. 2022

基于图神经网络的子图匹配符号算法

杨 欣，徐周波，陈浦青，刘华东

(桂林电子科技大学 计算机与信息安全学院，广西 桂林 541004)

摘 要：子图匹配是图数据分析中的基础问题，具有重要的研究意义。针对子图匹配求解算法存在大量冗余搜索的问题，提出了一种基于图神经网络的子图匹配符号算法。该算法利用图神经网络技术聚合节点的邻域信息，得到包含图局部属性和结构的特征向量，以该向量作为过滤条件得到查询图的节点候选集 C。此外，优化匹配顺序并利用符号 ADD 操作在数据图中构建 C 的各个候选区域，减少了子图枚举验证过程中的冗余搜索。实验结果表明，与 VF3 算法相比，该算法有效地提高了子图匹配的求解效率。

关键词：子图同构；图匹配问题；图神经网络；代数决策图；候选区

中图分类号：TP391.4　　**文献标志码**：A　　**文章编号**：1673-808X(2022)05-0391-07

Subgraph matching symbol algorithm based on graph neural network

YANG Xin, XU Zhoubo, CHEN Puqing, LIU Huadong

(School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract: Subgraph matching is a fundamental problem in graph data analysis and has important research significance. Aiming at the problem of a large number of redundant searches in the subgraph matching algorithm, a subgraph matching symbol algorithm based on graph neural network(SSMGNN) was proposed. The algorithm used the graph neural network technology to aggregate the neighborhood information of nodes, and obtained the feature vector containing the local attributes and structure of the graph, and used the vector as the filter condition to obtain the node candidate set C of the query graph. In addition, optimizing the matching order and using symbolic ADD operations to construct each candidate region of C in the data graph reduced redundant searches during subgraph enumeration verification. The experimental results show that, compared with the VF3 algorithm, the algorithm effectively improve the solving efficiency of subgraph matching.

Key words: subgraph isomorphism; graph matching problem; graph neural network; ADD; candidate region

图作为一种数据结构，可以有效刻画事物之间的关系，现实世界中的许多复杂问题都可以用图进行抽象表示。子图匹配问题作为图分析中最基本的问题之一，其目标是在数据图 g 中查找与查询图 q 同构的所有子图[1]，在图像检索、化学分子式检索、知识图谱查询和社交网络分析等领域有着广泛应用[2]。由于子图匹配问题属于 NP 难问题，随着数据规模的急剧增加，其求解的复杂度呈指数增长，因此广大研究者致力于扩大子图匹配问题的求解规模和提升求解效率。

在过去几十年，大量子图匹配算法被提出，其中基于回溯搜索的算法[3-13]更是被广泛研究。著名的 Ullmann 算法[3]于 1979 年提出，该算法基于回溯的树搜索在数据图 g 中枚举出与查询图 q 匹配的所有子图。由于 Ullmann 算法只采用了简单的剪枝策略，无法高效地减少搜索空间。VF2[4]算法在 Ullmann 算法的基础上，将节点的邻居信息作为约束条件，增强了剪枝效果，有效地减少了搜索空间。GraphQL 算法[5]提出以查询图的深度优先搜索生成树过滤数据图中的节点。Spath 算法[6]采用对查询图进行层次遍历并以每层节点的信息作为约束条件，引入复杂的结构与语义信息过滤数据图中的节点。

收稿日期：2022-04-15
基金项目：国家自然科学基金(61762027)；广西自然科学基金(2017GXNSFAA198172)
通信作者：徐周波(1976—)，女，教授，博士，研究方向为符号计算、智能规划、约束求解。E-mail：xzbli_11@guet.edu.cn
引文格式：杨欣，徐周波，陈浦青，等. 基于图神经网络的子图匹配符号算法[J]. 桂林电子科技大学学报，2022，42(5)：391-397.

这些算法虽然从不同角度增加剪枝过滤效果，从而改善了搜索效率，却很难在合理时间内对大规模数据图进行子图匹配求解。为了扩大求解规模，VF3 算法[7]在 VF2 的基础上依据查询图节点的匹配点在数据图中的出现概率计算匹配顺序，并引入分类概念，将节点按照度与标签进行分类，进一步减少了搜索空间。CFL-Match[8]算法提出分解查询图延迟笛卡尔积，以有效减少冗余的中间结果的产生，采用辅助数据结构 CPI 存储候选集中的部分连边。CECI[9]算法则将数据图划分为多个嵌入簇并进行并行处理，采用剪枝技术对嵌入簇反复修剪，得到辅助数据结构 CECI。虽然构造辅助数据结构提高了在大规模数据图中的搜索效率，却存在消耗大量的内存空间和预处理时间的问题。

为了平衡子图匹配求解算法的时间复杂度和空间复杂度，从增强过滤效果、优化匹配顺序和减少冗余搜索 3 方面出发，提出了基于图神经网络的子图匹配符号（symbol solving algorithm for subgraph matching based of graph neural network，简称 SSMGNN）算法。不同于 VFGCN 算法[14]，SSMGNN 算法通过引入图神经网络（graph neural networks，简称 GNN）[15-16]提取节点的特征向量，以该向量作为过滤条件来缩小节点候选集。其次，SSMGNN 算法在文献[9]的基础上，综合考虑了节点候选集的大小、节点度数以及待匹配节点的邻居中已匹配节点数等信息，给出了一种新的优化匹配顺序。同时，将引入的代数决策图（algebraic decision diagram，简称 ADD）[17]作为图的存储结构，采用符号 ADD 操作对子图匹配求解，在求解过程中，提出一种新的编码使得并行处理候选区[18]的划分，从而减少冗余搜索。因符号 ADD 是隐式存储，使得 SSMGNN 算法能在较小的存储空间中操作较大规模的图。

1 相关概念

定义 1 标签图 $G=<V,E,L,l>$，其中：V 表示图 G 的节点集合；E 表示图 G 中所有的边集；L 表示节点标签集合；l 表示标签映射函数：$V\to L$。

定义 2 给定数据图 $g=<V_g,E_g,L_g,l_g>$ 与查询图 $q=<V_q,E_q,L_q,l_q>$。图 q 与图 g 存在子图同构关系，当且仅当存在一个单射函数 $f:V_q\to V_g$ 满足：

1) $\forall u\in V_q,\exists f(u)\in V_g,l_q(u)=l_g(f(u))$；

2) $\forall(u,u')\in E_q,\exists(f(u),f(u'))\in E_g$.

定义 3 ADD 是表示基于变量序 $\pi:x_1<x_2<\cdots<x_n$ 的一族伪布尔函数 $f_i:\{0,1\}^n\to S$ 的一个有向无环图，它满足[19]：

1) S 为 ADD 代数结构的有限值域，且 $S\in\mathbf{Z}$。

2) ADD 中的节点分为根节点、终节点和内部节点 3 类。

3) 终节点集合记为 T。对任意 $t\in T$，均被标识为值域 S 中的一个元素是 $s(t)$。每个非终节点 u 具有四元组属性（f^u,n_{var},l,h），其中：f^u 表示节点 u 所对应的伪布尔函数；n_{var} 表示节点 u 的标记变量；l 表示节点 u 的 $n_{\text{var}}=0$ 时，节点 u 的 0-分支子节点；h 表示节点 u 的 $n_{\text{var}}=1$ 时，节点 u 的 1-分支子节点。

(4) 图中的每个节点 u 对应唯一一个函数 f^u。

(5) 图中任意一条从根节点到终节点的路径中，所有变量均按变量序 π 的顺序出现，且仅出现一次。

2 基于图神经网络的子图匹配符号算法

基于图神经网络的子图匹配符号算法由 3 个步骤组成，即生成候选集、生成匹配顺序和求解子图同构，算法的基本框架如图 1 所示。首先，从提升过滤效率的角度出发，引入图神经网络提取图节点的特征向量，并以该向量作为过滤条件为查询图生成候选集。其次，根据查询图节点的候选集、度以及查询图的层次结构等属性对匹配顺序进行优化。最后，使用 ADD 表示候选集与图，并结合 ADD 符号操作在数据图中以查询图的半径为范围并行构建候选区，并在候选区中采用回溯算法进行求解。

2.1 生成候选集

目前，大量的子图匹配算法可分为过滤与验证 2 个步骤，其过滤阶段将在数据图中探索查询图每个节点可能匹配的节点，这些匹配节点的集合称为该节点的候选集。在验证阶段，根据候选集的节点在数据图中进行搜索，从而得到子图匹配的可行解集。为了尽可能减少候选集中错误候选节点的数量，现有的子图匹配算法大多基于节点的标签与度信息对候选集进行过滤，却忽略了节点的局部邻域结构与属性信息。为此，用图神经网络对图的节点进行邻域信息聚合，得到节点特征向量，使得节点包含更多的局部信息，从而提升过滤效率。

首先利用图注意力网络[20]学习数据图与模式图的节点特征，得到其表示向量。其次，在文献[21]提出的子图预测函数的基础上，设计了函数 $p(\boldsymbol{x}_v,\boldsymbol{x}_u)$，该函数通过比较节点的表示向量中每个位置的数值大小来判断 2 个节点的邻域结构是否可能存在

图 1　SSMGNN 算法的基本框架

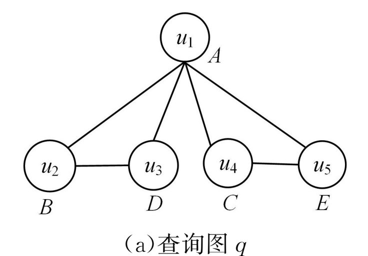
包含关系,如式(1)所示。

$$p(\boldsymbol{x}_v,\boldsymbol{x}_u)=\begin{cases}1, & \boldsymbol{x}_u[i]\leqslant \boldsymbol{x}_v[i], 1\leqslant i\leqslant d,\\ 0, & \text{其他。}\end{cases} \tag{1}$$

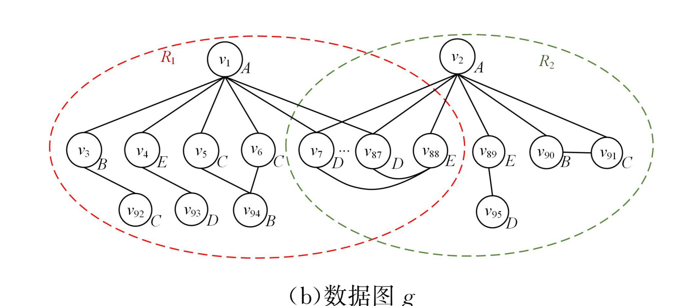
其中:$\boldsymbol{x}_v$ 为节点 v 的特征向量;$\boldsymbol{x}_v[i]$表示 $\boldsymbol{x}_v$ 的第 i 个值;d 为特征向量的维度。通过式(1)对数据图与查询图的每个节点的特征向量中每个位置的数值进行比较。若 $p(\boldsymbol{x}_v,\boldsymbol{x}_u)$为 1,表示节点 v 的节点邻域结构可能包含节点 u 的邻域结构,从而检查节点 u 与 v 的标签与度是否一致,一致则将节点 v 加入节点 u 的候选集。若 $p(\boldsymbol{x}_v,\boldsymbol{x}_u)$为 0,则说明 2 个节点的邻域信息不存在包含关系,从而节点 v 不能成为节点 u 的候选点。

2.2　确定匹配顺序

在子图匹配求解过程中,查询图节点匹配顺序的选择对搜索效率有着严重影响。如图 2 所示,查询图 q 中的每个节点在数据图 g 中的候选节点表示为 $(u_1,\{v_1,v_2\})$、$(u_2,\{v_3,v_{90}\})$、$(u_3,\{v_5,v_6,v_{91}\})$、$(u_4,\{v_7,v_8,\cdots,v_{91}\})$、$(u_5,\{v_4,v_{88},\cdots,v_{89}\})$。针对查询图 q,若给定匹配顺序 $P_1=\{u_4,u_1,u_5,u_2,u_3\}$,根据回溯搜索算法在数据图 g 中进行搜索求解,其搜索次数为 1 280 次,几乎需要遍历整个搜索空间。若匹配顺序为 $P_2=\{u_1,u_2,u_3,u_5,u_4\}$,其搜索次数为 175 次。由此可见,合适的匹配顺序能够有效减少搜索次数,从而提高搜索效率。



(a)查询图 q



(b)数据图 g

图 2　查询图 q 与数据图 g

因此,将综合考虑查询图节点的候选集、度数以及查询图的结构 3 个方面来对查询图匹配顺序进行优化。具体来说,优先匹配候选集小且度数大的节点,可降低中间结果的产生。由于在子图搜索的过程中会检查待匹配节点与已匹配节点的结构关系与查询图 q 是否一致,优先考虑与已匹配节点连边多的节点,也可提高搜索效率。因此,综合考虑节点候选集的大小、节点度数以及待匹配节点的邻居中已匹配数量,并选择两者之和的最大的节点作为下一个匹配节点,具体如式(2)所示。

$$\operatorname{order}(u)=\max\left(\frac{d(u)}{|C(u)|}+|N(u)\cap M|\right), \tag{2}$$

其中:$|\cdot|$ 表示取集合中元素的个数;$C(u)$表示节点 u 的候选集;$d(u)$表示节点 u 的度;$N(u)$为节点 u 的邻居节点集合;M 为已匹配节点的集合且初始值为空集;$|N(u)\cap M|$ 为节点 u 的邻居中已匹配的数量。第一个匹配节点称为根节点,因为选择根节点时 M 为空集,从而根据节点度数与节点候选点的数量的比值进行选择,优先选择两者比值大的节点作为根节点,并将根节点加入 M。下一个节点的选择,将会在节点度数与节点候选点的数量的基础上考虑该节点与 M 中节点的连边数量,若度数与节点候选点的数量的比值相同,则优先选择与 M 连边的节

点作为下一个节点。以此类推,确定所有节点的匹配顺序。

2.3 候选区

文献[9]指出在候选区中进行子图匹配能够有效减少搜索次数,并且在构造候选区的过程中可以进一步对候选集进行过滤。为了得到数据图中的所有同构子图,在构建候选区时,根节点的候选集中所有节点分别作为中心节点,在数据图中挖掘大小与查询图半径相同的子图作为候选区。然而,这一做法造成不同的候选区之间存在许多相同的节点和边,导致需要花费大量的空间来存储。ADD 是一种高紧凑、易操作的数据结构,可以对数据进行压缩存储,是克服存储容量限制的一种有效措施。因此,引入了符号 ADD,将其作为候选区的存储结构,并且可以并行地在数据图中进行搜索,得到所有候选区,有效解决了存储空间问题,且提高了构建候选区的效率。

2.3.1 ADD 表示

给定图 G,其节点数量 $|V|=n$,首先对图 G 的节点与边进行编码。节点编码可采用长度为 $m=\lceil \log_2 n \rceil$ 的二进制串 $X=(x_1,x_2,\cdots,x_m)$ 表示[22-23]。边代表节点与节点间的二元关系,因此边记为 $(X,Y)=(x_1,x_2,\cdots,x_m,y_1,y_2,\cdots,y_m)$,其中:$X$ 表示边的起始节点;Y 表示边的终节点。将数据图 g 的边集 E_g 转化为 ADD 表示,记为 $E_g(X,Y)$,查询图 q 边集的 ADD 表示记为 $E_q(X,Y)$。候选集的 ADD 表示与边相同,记为 $(X,Y)=(x_1,x_2,\cdots,x_m,y_1,y_2,\cdots,y_m)$,其中:$X$ 表示查询图的节点;Y 表示数据图的节点。候选集的 ADD 表示记为 $C(X,Y)$。

为了压缩存储空间并实现并行搜索,将所有候选区合并存储。因此,候选区的 ADD 表示添加变量 Z,以区分不同的候选区,记为 $(Z,X,Y)=(z_1,z_2,\cdots,z_m,x_1,x_2,\cdots,x_m,y_1,y_2,\cdots,y_m)$。其中:$Z$ 表示候选区区分标志;X 表示候选区中边的起始节点;Y 表示候选区中边的终节点。如图 2(b)所示,R_1 与 R_2 为数据图中的 2 个候选区,分别由根节点 u_1 的候选点 v_1 与 v_2 作为起始节点与该区的区分标志,数据图 g 的节点数 $n=97$,则 $m=7$,通过 Z 变量编码 0000001 与 0000010 来区分 R_1 与 R_2 候选区。

2.3.2 候选区构建

候选区的构建首先需要基于 2.1 节计算得到根节点,并通过根节点得到查询图的广度优先搜索树与查询图的半径 r。其次,根节点的候选点分别作为起始节点 v_s 并作为区别候选区的标志。然后,根据 v_s 在数据图中利用广度优先搜索构造以 r 为范围的候选区,在此过程中,每层候选区中的节点都会与查询图中该层节点的候选集做集合操作,以保留候选区中属于候选集的节点。构建候选区的伪代码如算法 1 所示。

算法 1 构建候选区(ADDPartiton)

输入:数据图边集 $E_g(X,Y)$,查询图边集 $E_q(X,Y)$,候选集 $C(X,Y)$,查询图半径 r,根节点 root(X)。

输出:候选区 $R(Z,X,Y)$。

1. $C_{root}(X)$=GetRootCand($C(X,Y)$,root(X));
2. $R_{tmp}(Z,X)=\emptyset$,$R(Z,X,Y)=\emptyset$;
3. $R_{tmp}(Z,X)$=RegionMark($C_{root}(X)$);
4. $T_g=R_{tmp}(Z,X)$,T_q=root(X),$L_{q_0}(X)$=root(X);
5. for($i=0$;$i<r$;i++)
6. $\quad$ $L_n(Z,X)$=GetLayerNode($L_{q_0}(X)$,$E_q(X,Y)$,T_q,$R_{tmp}(Z,X)$,$C(X,Y)$,$E_g(X,Y)$);
7. $\quad$ $R(Z,X,Y)$=UpdateRegion($L_n(Z,X)$,$R_{tmp}(Z,X)$,$E_g(X,Y)$,$R(Z,X,Y)$);
8. $\quad$ UpdateParameter(T_g,T_q,$L_{q_i}(X)$,$L_n(Z,X)$);
9. Return $R(Z,X,Y)$。

算法 1 的第 1 行函数 GetRootCand 是将根节点 root(X)与查询图节点的候选集 $C(X,Y)$做 ADD 符号操作,返回根节点的候选集 $C_{root}(X)$。第 3 行函数 RegionMark 是通过符号操作构造以 $C_{root}(X)$中的每个节点为起始点与区分候选区节点的候选区 $R_{tmp}(Z,X)$。第 5~8 行是一个以查询图的半径为范围的 for 循环,根据广度遍历搜索,在数据图中一层一层地往下搜索并更新候选区。其中第 6 行函数 GetLayerNode 是为了得到候选区的第 $i+1$ 层的节点,首先通过查询图操作得到第 $i+1$ 层的节点并得到该层节点的候选集,用该候选集过滤数据图操作得到第 $i+1$ 层的节点,从而得到候选区的第 $i+1$ 层的节点 $L_n(Z,X)$。第 7 行函数 UpdateRegion 为更新增加候选区的边集,将候选区 $R(Z,X,Y)$中第 i 层节点与 $L_n(Z,X)$节点的连边和每个候选区中 $L_n(Z,X)$节点间的连边加入候选区 $R(Z,X,Y)$。第 8 行函数 UpdateParameter 为更新数据图在不同候选区中已经遍历过的节点、查询图已经遍历的节点、已经分区待操作的数据集中的节点和查询图中待操作的节点,更新后进入下一次 for 循环。第 9 行返回候选区 $R(Z,X,Y)$。

2.4 SSMGNN 算法

结合符号 ADD 技术构建候选区并进行求解,给

出基于图神经网络的子图匹配符号算法(SSMGNN)。该算法主要步骤为:第 1 步产生候选集,利用 GNN 对图节点邻域信息聚合并得到特征向量,使用该向量计算得到查询图候选点。第 2 步优化匹配顺序,为后续的搜索匹配做准备。第 3 步构建候选区,在数据图中利用符号 ADD 操作构建候选集的各个候选区域。第 4 步子图匹配求解,结合 ADD 操作与回溯搜索进行求解。算法 SSMGNN 的伪代码如算法 2 所示。

算法 2 SSMGNN

输入:数据图 $g=<V_g,E_g,L_g,l_g>$,模式图 $q=<V_q,E_q,L_q,l_q>$数据图中每个顶点的节点嵌入 $\boldsymbol{X}_v$,数据图边集的 ADD 图 $E_g(X,Y)$。

输出:子图匹配的所有解 solutions。

```
1.  X_u = NodeEmbedding(q);
2.  C(u) = ComputeCand(X_u, X_v, g, q);
3.  r, order, root ← Order(q, C(u));
4.  order(X,Y), E_q(X,Y), C(X,Y), root(X) ←
    CreateADD(order, E_q, C(u), root);
5.  R(Z,X,Y) = ADDPartition(E_g(X,Y), E_q(X,
    Y), C(X,Y), r, root(X));
6.  solution = ∅;
7.  C_root(X) = GetRootCand(C(X,Y), root(X));
8.  for v_root(X) ∈ C_root(X)
9.      R_i(X,Y) = ChooseRegion(v_root(X), R(Z,
        X,Y));
10.     if CheakRegion(C(X,Y), R_i(X,Y)) == ∅
11.         break;
12.     else
13.         Match(order(X,Y)), R_i(X,Y), v_root(X),
            solutions);
14. Return solutions。
```

算法 2 第 1 行函数 NodeEmbedding 用于对查询图的节点进行邻居信息聚合并表示为特征向量。第 2 行函数 ComputeCand 使用式(1)计算数据图与查询图的节点特征,得到查询图节点在数据图中的候选集。第 3 行 Order 函数用于优化模式图的节点匹配顺序。通过候选集和查询节点的度等信息确定查询图的根节点 root。以 root 为起始点对查询图进行广度遍历搜索,并对每层的节点进行排序,该排序按照已匹配节点的连边、候选集大小与度等条件得到最终的匹配顺序。第 4 行函数 CreateADD 是将排序后的节点,查询图的边集、候选集与根节点转化为 ADD 图表示。第 5 行函数 ADDPartition 使用 ADD 操作在数据集构建多个候选区 $R(Z,X,Y)$,具体过程在算法 1 中给出。第 7 行函数 GetRootCand 通过 ADD 符号操作返回 root 的候选集 $C_{\mathrm{root}}(X)$。第 8~14 行是以 $C_{\mathrm{root}}(X)$的节点为起始节点的每个候选区进行子图同构匹配。其中第 9 行函数 ChooseRegion 将选择 $C_{\mathrm{root}}(X)$中的一个节点与候选区 $R(Z,X,Y)$做符号操作,并返回其中的一个候选区 $R_i(X,Y)$;第 10~13 行 CheakRegion 函数是检查 $R_i(X,Y)$的可行性,查看每个查询点的候选集在 $R_i(X,Y)$中是否为空,若为空,则说明该区域无解,需要重新选择一个候选区,否则,使用第 13 行函数 Match 进行求解,在 $R_i(X,Y)$中按照 order 的匹配顺序对变量进行赋值,采用回溯搜索算法寻找可行解并将其加入解集;第 14 行返回所有可行解。

3 实验结果及分析

实验环境为 Core(TM) i5-1038NG7 CPU @ 2.00 GHz,16 GiB 内存,操作系统为 Windows 10 64 位,编译语言为 Python 和 C/C++。实验使用的 2 个公开数据集分别是 human 和 yeast。其中 human 为人类蛋白质相互作用的数据图,由 86 282 条边、4 674 个节点和 44 个不同的标签组成的无向图;Yeast 数据集包含了 12 519 条边、3 112 个节点与 71 个不同的标签。对于每个数据集,构造 4 种不同的查询集,分别为 Q_4、Q_8、Q_{16}、Q_{32},其中 Q_i 表示包含 i 个节点的查询图的查询集。每个查询集包含 50 个具有相同数量节点的连通图。由于 VF3 算法是单机上非常高效的算法,其性能优于 VF2[4]、L2G[24]、LAD[11] 和 RI[10] 算法,因此本实验将与 VF3 算法进行比较。实验结果中的平均时间为每组查询图得到子图同构所有解计算的总时间与查询数量的商,平均时间作为评估指标更能体现子图同构算法的时间效率。

实验结果如图 3 所示,SSMGNN-yeast 与 VF3-yeast 分别表示 SSMGNN 算法与 VF3 算法在 yeast 数据集上的平均运行时间;同理,SSMGNN-human 与 VF3-human 分别表示 2 种算法在 human 数据集上的平均运行时间。通过实验对比可知:

1)当查询集为 Q_4 时,VF3 算法在 2 个数据集上的平均求解时间较少,优于本算法,其原因是在算法执行过程中,构建 ADD 需要消耗一定的时间,并且 SSMGNN 算法利用图神经网络提取节点的邻域信息作为过滤条件,在模式图较小的时候,其邻域包含信息较少,从而过滤效果并不明显。然而,随着查询图规模的增加,SSMGNN 算法的平均时间性能明显优于 VF3 算法,且时间增幅小于 VF3 算法,这表明本算法在处理较大规模的查询图时更具有优势。尽

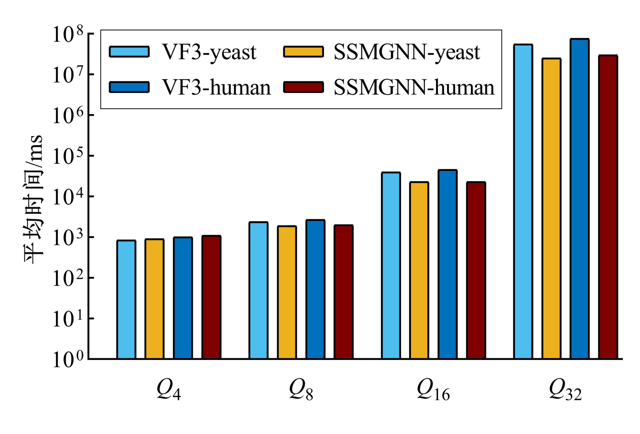


图 3 human 与 yeast 数据集的实验结果

管当查询图规模增加时，2 个算法的时间复杂度都明显增加，但是本算法通过优化匹配顺序与增加节点邻域信息作为过滤条件，有效减小了搜索空间，使得时间效率得到提升。

2）VF3 算法在 human 数据集的求解时间比在 yeast 数据集上耗时更长。其原因是，human 数据集是稠密图并且标签更少，VF3 算法在 human 数据集需要花费更多的时间进行搜索求解。但是，SSMGNN 算法在稠密和稀疏数据集上的求解时间相当。由此表明，本算法不仅在稀疏图上能够高效求解，并且在稠密图上的求解效率远超于 VF3。

查询集的聚合时间如表 1 所示，对于不同规模下的查询图，邻域信息聚合的平均时间都在 0.001 2 s 左右，由此可见，图神经网络在 4～32 个节点的规模下都能够花费非常少的时间对节点的邻域信息进行特征提取。

表 1 查询集的邻域聚合时间

查询集规模	数量	总时间/s	平均时间/s
Q_4	50	0.054 8	0.001 1
Q_8	50	0.055 9	0.001 1
Q_{16}	50	0.065 4	0.001 3
Q_{32}	50	0.689 0	0.001 4

在实验的过程中发现，对于邻域信息聚合时，并不是聚合越多的邻域信息其过滤效果越好，在聚合时所选择的 k 值（节点的 k 步邻居）为 2 时效果最好。原因是 human 与 yeast 数据集中的查询图的半径范围几乎不超过 3，当 k 值越大时，会使得所有节点的邻域信息趋于一致，从而降低其过滤效果。

5 结束语

提出了一种结合图神经网络与符号代数决策图解决子图同构问题的算法。该算法利用图神经网络聚合节点邻居信息，提高了过滤候选节点的效率，并且基于候选集、度与子图的结构等特征提出了一种优化的匹配顺序。其次，构建了关于数据集、查询图与候选集的 ADD 图，使用符号 ADD 操作实现了在单机上并行划分候选区并进行回溯求解。实验结果表明，该算法有效提高了子图同构问题的求解效率。在未来的工作中，将研究如何使用符号 ADD 并行求解子图同构问题，进一步提高在单机上的求解效率。

参考文献：

[1] SUN S X, LUO Q. In-memory subgraph matching: an in-depth study[C]//Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. Portland, Oregon: ACM Press, 2020: 1083-1098.

[2] HAN M, KIM H, GU G, et al. Efficient subgraph matching: harmonizing dynamic programming, adaptive matching order, and failing set together[C]//Proceedings of the 2019 International Conference on Management of Data. Amsterdam: ACM Press, 2019: 1429-1446.

[3] ULLMANN J R. An algorithm for subgraph isomorphism[J]. Journal of the ACM, 1976, 23(1): 31-42.

[4] CORDELLA L P, FOGGIA P, SANSONE C, et al. A (sub) graph isomorphism algorithm for matching large graphs[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26(10): 1367-1372.

[5] HE H, SINGH A K. Graphs-at-a-time: query language and access methods for graph databases[C]//Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver Canada: ACM Press, 2008: 405-418.

[6] ZHAO P X, HAN J W. On graph query optimization in large networks[J]. Proceedings of the VLDB Endowment, 2010, 3(1/2): 340-351.

[7] CARLETTI V, FOGGIA P, SAGGESE A, et al. Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with VF3[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 40(4): 804-818.

[8] BI F, CHANG L J, LIN X M, et al. Efficient subgraph matching by postponing cartesian products[C]//Proceedings of the 2016 International Conference on Management of Data. San Francisco: ACM Press, 2016: 1199-1214.

[9] BHATTARAI B, LIU H, HUANG H H. Ceci: compact embedding cluster index for scalable subgraph matching[C]//Proceedings of the 2019 International Conference on Management of Data. Amsterdam Neth-

erlands:ACM Press,2019:1447-1462.
[10] BONNICI V,GIUGNO R,PULVIRENTI A,et al. A subgraph isomorphism algorithm and its application to biochemical data[J]. BMC Bioinformatics,2013,14(7):1-13.
[11] SOLNON C. All different-based filtering for subgraph isomorphism[J]. Artificial Intelligence,2010,174(12-13):850-864.
[12] SHANG H C,ZHANG Y,LIN X M,et al. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism[J]. Proceedings of the VLDB Endowment,2008,1(1):364-375.
[13] REN X G,WANG J H. Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs[J]. Proceedings of the VLDB Endowment,2015,8(5):617-628.
[14] 徐周波,李珍,刘华东等. 基于邻居信息聚合的子图同构匹配算法[J]. 计算机应用,2021,41(1):43-47.
[15] SCARSELLI F,GORI M,TSOI A C,et al. The graph neural network model[J]. IEEE Transactions on Neural Networks,2008,20(1):61-80.
[16] WU Z H,PAN S R,CHEN F W,et al. A comprehensive survey on graph neural networks[J]. IEEE Transactions on Neural Networks and Learning Systems,2020,32(1):4-24.
[17] BAHAR R I,FROHM E A,GAONA C M,et al. Algebric decision diagrams and their applications[J]. Formal Methods in System Design,1997,10(2):171-206.
[18] HAN W S,LEE J,LEE J H. Turboiso: towards ultrafast and robust subgraph isomorphism search in large graph databases[C]//Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. New York:ACM Press,2013:337-348.
[19] 古天龙,徐周波. 有序二叉树决策图及应用[M]. 北京:科学出版社,2009:92-111.
[20] VELICKOVIC P,CUCURULL G,CASANOVA A,et al. Graph attention networks[EB/OL]. (2018-02-04)[2022-04-02]. https://arxiv. org/pdf/1710. 10903. pdf.
[21] YING R,WANG A,YOU J X,et al. Neural subgraph matching[EB/OL]. (2020-10-27)[2022-04-02]. https://arxiv. org/pdf/2007. 03092. pdf.
[22] 刘桂珍,徐周波. 基于符号 OBDD 的子图同构约束求解算法[J]. 桂林电子科技大学学报,2019,39(5):357-362.
[23] 徐周波,古天龙,赵岭忠. 网络最大流问题的一种新的符号 ADD 求解算法[J]. 通信学报,2005,26(2):1-8.
[24] ALMASRI I,GAO X,FEDOROFF N. Quick mining of isomorphic exact large patterns from large graphs[C]//2014 IEEE International Conference on Data Mining Workshop. Piscataway,NJ:IEEE Press,2014:517-524.

编辑:梁王欢